FIB UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Multimodal Neural Machine Translation System

Author: Zhiwen Tang

Director: Marta Ruiz Costa-jussà

Co-Director: Lluís Padró Cirera

May 27, 2016

Abstract

With the development of society and globalization, communication among people coming from different cultural background is more and more common where the language becomes the major obstacle that hamper the exchange of information. People need an automatic translator that can help them with communication. As the development internet where multimedia text is becoming the mainstream, a multimodal translator is more desired than before.

Deep learning has been proven to be successful in solving hard problems in Computer Vision and Speech Recognition. Inspired by these progress, I try to use deep learning techniques to complete the task of multimodal translation. In the task, an image caption in the target language (German) needs to be generated based on the raw image and one or more image description in the source language (English).

In this report, I proposed a set of methods to complete the task by integrating a Neural Machine Translator, a Neural Image Caption Generator and an Evaluation Module. The integration of these methods not only completed the task, but also made improvements.

The Neural Machine Translator will translate captions directly with an Encoder –Decoder architecture with soft attention mechanism, where a bidirectional Recurrent Neural Network will encode the source sentence and an attention based Recurrent Neural Network will decode it into the target sentence.

The Neural Image Caption Generator will generate the caption based on the based on the image with a Long Short-Term Memory based sentence generator which takes the features extracted from a Convolutional Neural Network.

The Support Vector Machine-based Evaluation module will choose the best sentence from the sentences generated by two subsystems. The evaluation module will assess the quality of sentences considering correctness and fluency at the same time.

Keywords: Machine Translation, Multimodal, Convolutional Neural Network, Recurrent Neural Network, Support Vector Machine

Index

1 Introduction

1.1 Background

With the development of economics and society, the communications among people from different countries with different cultural background has become more and more common, where language is a major obstacle. People are desiring an automatic translator which can help them understand their partners in business and research who speak or write another language. Technology giants, such as Google and Microsoft, also noticed that strong demand, which motivated them developing commercial machine translators, for example, Bing Translator[1]. These commercial machine translators use various technologies and performs well on text translation. However, blank still exists in multimodal machine translation.

Deep learning is a very hot topic today. Aided by deep learning techniques, computer scientists have solved many complicated problems in computer vision and speech recognition with different architecture of deep neural networks. Inspired by those great progress, researchers are using deep learning techniques to complete machine translation, which brings birth to neural machine translation.

In this report, a method is proposed to complete the translation of multimodal text with neural network. This project is set in the framework of WMT16[2] task, which consists in generating a German caption regarding a picture given the picture itself and its one or more English captions. By utilizing the techniques of image caption generation and machine translation, I completed this task successfully. I also show that an SVM-based sentence re-ranker can improve the performance of the whole system greatly by considering the accuracy and fluency at the same time. Related concepts and algorithms will be introduced later.

1.2 Concepts

Natural Language Processing (NLP) is a crosscutting area of Computer Science and Linguistics. Research of Natural Language Processing concentrates on the interaction between human and computer, namely, the input and output in natural language. Research focusing on the input direction is also called Natural Language Understanding, which makes computers to transform the input from human in natural language via text or voice to the way it expresses its own knowledge. Research addressing the output problem is also called Natural Language Generation, outputting data inside computer in human natural language.

Machine translation (MT) is a subfield of Natural Language Processing. The aim of machine

[1] https://www.bing.com/translator
[2] http://www.statmt.org/wmt16/multimodal-task.html

translation is to translate text from a language (source language) to another language (target language) by the computer itself without the involvement of human.

Currently, rule-based methods, statistical-based methods, hybrid-based methods are three most widely used machine translation methods (Antony 2013):

Rule-based Method: Rule-based machine translation employs the translation rules between two languages found by linguists. The biggest advantage of rule-based machine translation is, every rule is explicit, making it easier to implement, which also brings the disadvantage, in many occasions, translation rules between two languages are difficult to be summarized by human.

Statistical-based Method: Statistical machine translation builds model on the base of bilingual text corpora (like EUROPARL (Koehn 2005)) using statistical methods. The core of this kind of method is making the machine discover the translation rules automatically by learning the bilingual text. Usually, Statistical machine translation relies on the large corpus, which are still rare. **Neural machine translation** is one kind of Statistical machine translation.

Hybrid-based Method: Hybrid-based machine translation, which has better efficiency, is a combination of Rule-based method and Statistical-based method. In some case, rules are used to preprocess the input data and post-process the output data of statistical-based system. In other cases, rules are used first and statistical method will be used later to refine the results.

Syncretic texts draw on several systems of semiotic resources (including but not limited to language, image, music, color or perspective) is often referred as **multimodal texts** (González 2014). For example, in Figure 1, the image, where a girl is playing with guitar, and the sentence beside, “a girl is playing with guitar”, make up multimodal texts.

Multimodal machine translation is a special class of machine translation with multimodal input or output. Compared with traditional machine translation, multimodal machine translation takes more than text as input, it also gains features from sound or image while generating the corresponding translated text. Multimodal neural machine translation is still an emerging area.

A girl is playing with guitar

Figure 1 Multimodal texts

1.3 Aim

The aim of this project is to build a multimodal neural machine translation system, which takes an image and one or more captions in the source language (English), to generate a caption in the target language (German). After doing the research about this topic, this task is decomposed into three objectives.

Objective 1: to build a neural machine translator, which takes sentences in a source language as input, to generate sentences in a target language

Objective 2: to build a neural image caption generator, which takes an image as input to output corresponding caption in the target language

Objective 3: to build a sentence re-scorer, which will utilize a language model, considering the accuracy and fluency of sentences at the same time, so as to choose the best sentence.

The final goal cannot be achieved until all three objectives have been satisfied.

1.4 State of Art

In this part, the state of art involving those three objectives will be introduced.

1.4.1 Neural Machine Translation

In general, the architecture of neural machine translation can be divided into two classes, Encoder-Decoder Model and Attention Model.

Encoder-Decoder model: The most basic Encoder-Decoder models consist of an encoder, a Recurrent Neural Network (RNN) or a Convolutional Neural Network (CNN), and a decoder, a Recurrent Neural Network (RNN). Models of this type will first be sent into the encoder, transforming the source sentence into a length-fixed vector. Then, this vector will be sent into the decoder, where it will be decoded into the sentence in the target language (Auli, Galley et al. 2013, Kalchbrenner and Blunsom 2013, Cho, Van Merriënboer et al. 2014, Sutskever, Vinyals et al. 2014). The major disadvantage of Encoder-Decoder model is that its performance decreases rapidly as the length of sentence and number of unknown words increase (Cho, van Merriënboer et al. 2014).

Attention-based Model: In order to improve the weakness of Encoder-Decoder model, attention-based model came into being. The architecture of Attention-based models is very similar to that of Encoder-Decoder models. However, in Attention-based models, supported by Recurrent Neural Network, source sentence is represented as a series of vectors (instead of only one vector in

Encoder-Decoder models). After that, a gating neural network will dynamically align the sentences simultaneously, and another Recurrent Neural Network will generate the sentences in the target language (Bahdanau, Cho et al. 2014, Luong, Pham et al. 2015).

Experiments show that Attention-based models are more effective than Encoder-Decoder models. Attention-based model can achieve comparable results in fewer parameters and smaller training set (Cho, Memisevic et al. 2015). This benefit is brought by the dynamic aligning mechanism, which avoids representing the whole sentence in a fixed-length vector (Sutskever, Vinyals et al. 2014).

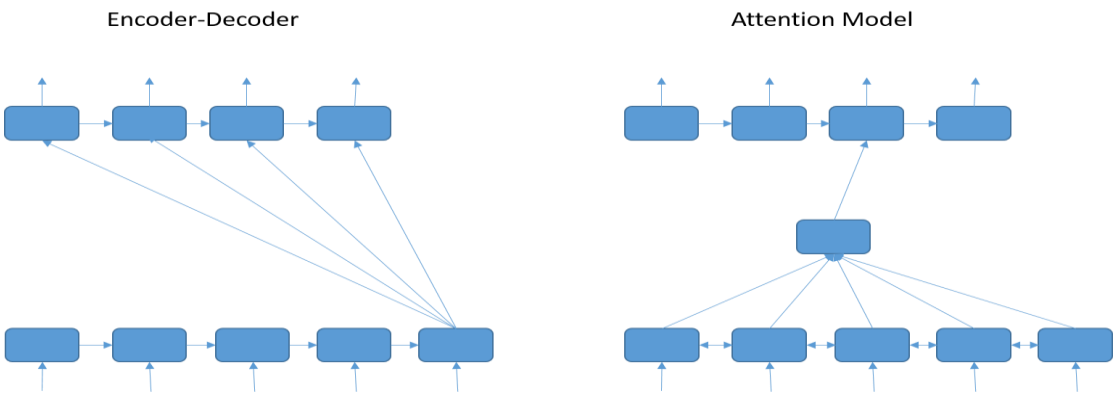


Figure 2 Two types of neural machine translation

1.4.2 Image Captioning

Generally, the image captioning methods can be categorized into three classes (Kiros, Salakhutdinov et al. 2014).

Template-based methods: This type of methods mainly involves how to fill the templates (such as triplets). Template-based methods will identify the objects from the image and extract their spatial relationship. Information extracted will be used as fill in the templates (Farhadi, Hejrati et al. 2010, Li, Kulkarni et al. 2011, Yang, Teo et al. 2011, Mitchell, Han et al. 2012, Kulkarni, Premraj et al. 2013). These kind of methods are more robust with higher accuracy. However, sentences generated are not as fluent and natural as sentences written by human.

Composition-based Methods: The core of this kind of methods is to utilize existing image-caption datasets, like Flicker30k (Young, Lai et al. 2014), Microsoft COCO (Lin, Maire et al. 2014). Components of related captions are first extracted from the dataset and then composed together to generate new image captions (Kuznetsova, Ordonez et al. 2012, Kuznetsova, Ordonez et al. 2014). The advantage of this kind of methods is that, compared with Template-based methods, they can be used more widely and sentences generated are more natural and closer to those written by human.

Neural Network Methods: Neural Network Methods generate sentences by sampling from Neural Language Model. In this kind of method, first, image will be sent into deep neural network to extract features. After that, feature vectors extracted will be used by Recurrent Neural Network to

generate image captions (Kiros, Salakhutdinov et al. 2014, Mao, Xu et al. 2014). Currently, Neural Network Methods and Composition-based based Methods can achieve the state-of-art performance. For instance, Xu et al achieve the best on Flicker30k dataset with CNN + LSTM architecture (Xu, Ba et al. 2015).

1.4.3 Language Models

Sentences output from neural machine translator or image caption generator can be re-scored with a language model, which will evaluate the accuracy and fluency of the sentence. With the help from language model, sentence with the best quality will be chosen as the final output of the translation.

Usually, language models assume that the probability of the appearance of the next word is conditioned on the previous words and the probability of the appearance of sentence T is the product of the probabilities about the appearance of all the words in sentence T.

$$p(T) = p(w_1 w_2 \dots w_m) = \prod_{i=1}^{m} p(w_i | w_1 \dots w_{i-1})$$
$$= p(w_1) * p(w_2|w_1) * p(w_3|w_1 w_2) * \dots * p(w_m|\, w_1 w_2 \dots w_{m-1}) \quad (1)$$

However, (1) is hard to be implemented. So, researchers tried to use different methods to get approximation or the accurate value of (1).

N-gram model: n-gram model made some simplifications: it is based on the hypothesis that the probability of n-th word is conditioned only on the previous $\mathrm{n}-1$ words, meaning the generation of a sentence can be seen as a Markov process.

$$p(T) = p(w_1 w_2 \dots w_m) = \prod_{i=1}^{m} p(w_i | w_{i-n+1} \mathrm{w_{i-n+2}} \dots \mathrm{w_{i-1}}) \quad (2)$$

Probabilities can be obtained by counting large corpus. The most common n-gram language models are Bigram model (n=2) and Trigram model (n=3).

Neural Language Model: with the rise of neural network, researchers are also trying to build the language model with neural network. Bengio et al proposed to use a three-layer Feedforward Neural Network to get n-gram model with better performance (Bengio, Ducharme et al. 2003). And Recurrent Neural Network is also used to get the language model with fewer parameters (Mikolov 2012).

1.5 Thesis Structure

This project report will be organized in the following structure:

1 Introduction: background and basic information about this project, as well as the state of art

2 Program Management: initial plan and final plan of this project, as well as the reasons of derivations

3 Basic Theories: theories applied in this program

4 Program Implementation: the architecture and implementation details of the program

5 Experiments: experiment result on the test set and samples

6 Conclusion: evaluation of the result of this program according to the initial goal, as well as future work

2 Program Management

In this part, I will explain the details about the initial plan and the final plan, as well as the reasons behind the derivations. In the meantime, an assessment about the sustainability will also be given.

2.1 Project Analytics

2.1.1 Task Decomposition

The goal of this bachelor final project is to build a multimodal neural machine translation system, which takes an image and its caption in the source language (English) as input, to generate its corresponding caption in the target language (German). In time order, the whole project can be divided into three stages, **planning**, **development** and **final stage**. The final stage and either of the other two stages cannot be done in parallel. While it is impossible to start the all the work in development stage before the planning stage is over, some parts of the development stage can be done in prior before the end of planning stage, which can save a lot of time.

2.1.2 Stage 1 Planning

This stage is the first stage of the whole project. Everything in this stage has been covered by GEP course. I made a thorough plan about my Bachelor Final Project in this stage, the following parts is included:

- Definition of Context and Scope of this project
- Temporal Planning
- Economic Management and Sustainability

2.1.3 Stage 2 Development

In this stage, I built my multimodal neural machine translation system.

This task of this project can be viewed from two different angles. One the one hand, it can be seen as a caption generation task supported by caption's features in the source language. On the other hand, it can also be regarded as a machine translation task aided by features from the image. From this point of view, the task of this stage can be divided into three different subtasks.

- Subtask 1. Translating caption from source language to target language
- Subtask 2. Generating caption in the target language from image

- Subtask 3. Merging the results and output

Subtask 1 and subtask 2 can be completed simultaneously while subtask 3 cannot be done until subtask 1 and subtask 2 have been finished.

Subtask 1

In this subtask, I built the first baseline system (which is called neural machine translator), which take the caption text in the source language as input and output the corresponding text translation. Neural Machine Translator is built on previous work.

Subtask 2

The second baseline system (which is called Neural Image Caption Generator) is built in this stage. Neural Image Caption Generator generates the captions in the target language given the raw images. Similarly, Neural Image Caption Generator is also based on others' previous work.

Subtask 3

This part is the most important part of this project. Given the image and its caption in the source language, *baseline system 1* and *baseline system 2* may generate two different results. This submodule is used to evaluate two results and generate the final output considering the fluency and accuracy and the same time.

In this stage, I used the method of Agile Programming, building runnable system iteratively and quickly. The improvement is based on the running result on the test dataset.

2.1.4 Stage 3 Final Stage

In this stage, my job is to make sure all the components are working perfectly, completing all the documents and report required and preparing for the final defense.

2.2 Resources Used

The following resources, including hardware and software are used in this project.

2.2.1 Hardware

Linux Server (equipped with GPU)
Lenovo ThinkPad E440

Figure 3 Hardware Used

2.2.2 Software

Linux Fedora
Microsoft Windows 10
Pycharm 5
XShell 5
XFtp 5
Microsoft Office 2013
Adobe Reader

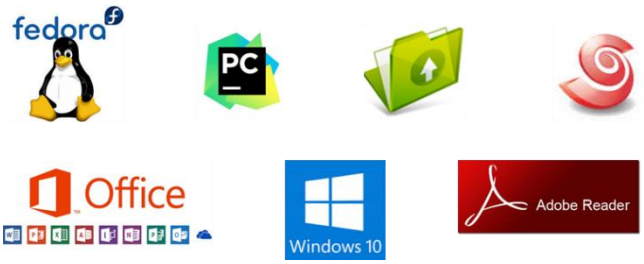


Figure 4 Software Used

2.3 Initial Plan

Initially, I planned to complete the whole project in about 2 months and defend it in April. Based on this assumption, I made my initial plan.

The initial plan includes the time plan and estimated budget. The final plan is different from the initial one because of multiple reasons, which will be explained in the 2.4 Reasons of derivations.

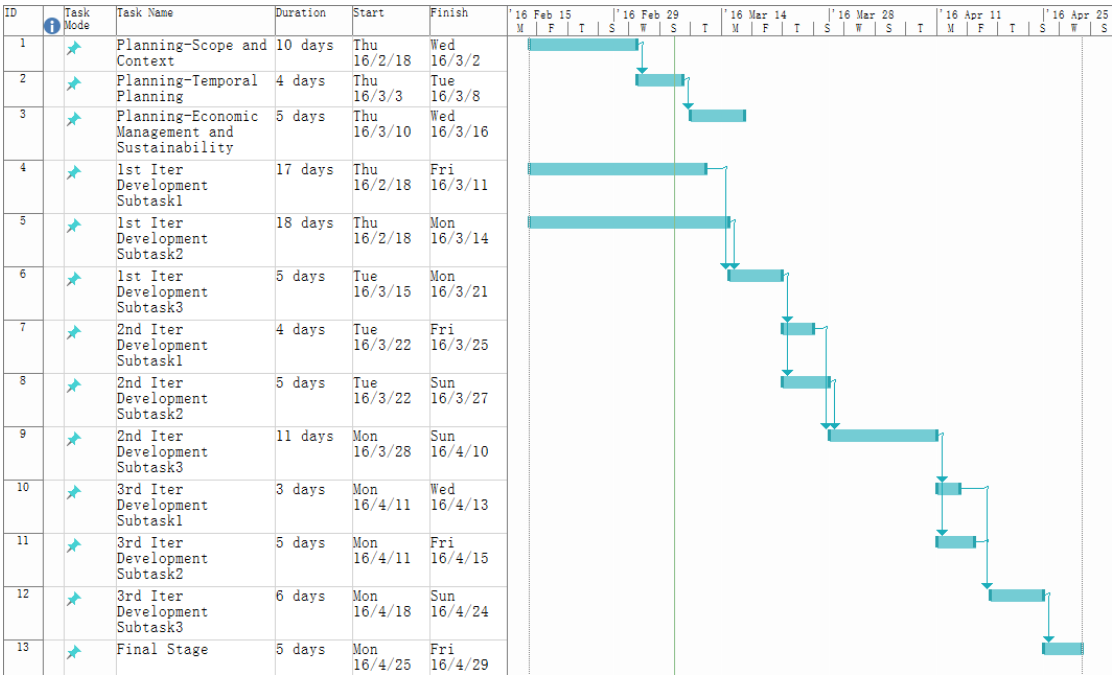
2.3.1 Initial Time Plan

The initial time plan is based on the preliminary research about this project, time expected for every stage and task is estimated according to the extent of difficulties.

Stage	Estimated Time (hours)
Planning Stage	80
Development Stage: Subtask 1	80
Development Stage: Subtask 2	100
Development Stage: Subtask 3	180
Final Stage	60
Total	500

Table 1 Initial Estimated Time

Gantt Chart of the initial plan is also shown in Figure 5.



ID	Task Mode	Task Name	Duration	Start	Finish
1		Planning-Scope and Context	10 days	Thu 16/2/18	Wed 16/3/2
2		Planning-Temporal Planning	4 days	Thu 16/3/3	Tue 16/3/8
3		Planning-Economic Management and Sustainability	5 days	Thu 16/3/10	Wed 16/3/16
4		1st Iter Development Subtask1	17 days	Thu 16/2/18	Fri 16/3/11
5		1st Iter Development Subtask2	18 days	Thu 16/2/18	Mon 16/3/14
6		1st Iter Development Subtask3	5 days	Tue 16/3/15	Mon 16/3/21
7		2nd Iter Development Subtask1	4 days	Tue 16/3/22	Fri 16/3/25
8		2nd Iter Development Subtask2	5 days	Tue 16/3/22	Sun 16/3/27
9		2nd Iter Development Subtask3	11 days	Mon 16/3/28	Sun 16/4/10
10		3rd Iter Development Subtask1	3 days	Mon 16/4/11	Wed 16/4/13
11		3rd Iter Development Subtask2	5 days	Mon 16/4/11	Fri 16/4/15
12		3rd Iter Development Subtask3	6 days	Mon 16/4/18	Sun 16/4/24
13		Final Stage	5 days	Mon 16/4/25	Fri 16/4/29

Figure 5 Gantt chart of Initial Plan

2.3.2 Initial Estimated Budget

In order to complete this project, human resource, hardware resource and software resource are used. Considering the amortizations, the initial estimated budget is shown in Table 2.

Resource	Price(Euro)	Useful Life	Amortization(Euro)
Linux Server(with GPU)	20181.00	8 years	630.66
Lenovo Thinkpad E440	1225.00	5 years	61.25
Total	3729.00		691.91

Table 2 Initial Hardware Budget

Resource	Price(Euro)	Useful Life	Amortization(Euro)
Linux Fedora	0.00	3 years	0.00
Microsoft Windows 10	99.95	3 years	8.33
PyCharm 5 Community Edition	0.00	3 years	0.00
XShell 5 (Free for school/home use)	0.00	3 years	0.00
XFtp 5 (Free for school/home use)	0.00	3 years	0.00
Microsoft Office 2013	179.90	3 years	15.00
Adobe Reader	0.00	3 years	0.00
Total	279.85		23.33

Table 3 Initial Software Budget

In order to cut budget, I decided to use Open Source software or free software as much as possible given the performance of open source software or free software is as good as that of commercial billing software.

Although this project is going to be completed by only one person, however, this person (which is me) needs to play as different roles at different time. In the whole project, this person is a project manager at first (just for planning stage), and then, he becomes the Software Programmer who is responsible for programming at every iteration of development. At the end of every iteration, he becomes the Software Tester to complete the test of every version of software.

Resource	Working Time(hours)	Price (euros/hour)	Cost(euros)
Program Manager	80	50.00	4000.00
Software Programmer	300	40.00	12000.00
Software Tester	120	35.00	4200.00
Total			20200.00

Table 4 Initial Human Resource Budget

Combining the budget on all the resources that will be used, the total initial budget of this program is shown in Table 5.

Resource	Budget
Hardware Resource	691.91
Software Resource	23.33
Human Resource	20200.00
Total	20915.24

Table 5 Initial Total Budget

2.3 Final Plan

Due to the limited time, huge workload and unexpected difficulties, the project is not completed as scheduled before, in fact, this project is completed at the middle of May, which brings the increase in human resource cost as well as the amortizations on the software and hardware resources.

2.3.1 Final Timetable

The actual cost of time of this project is shown in Table 6.

Stage	Time Cost (hours)

Planning Stage	80
Development Stage: Subtask 1	60
Development Stage: Subtask 2	200
Development Stage: Subtask 3	200
Final Stage	60
Total	600

Table 6 Actual Cost of Time

The Gantt chart of the final time table is shown in Figure 6.

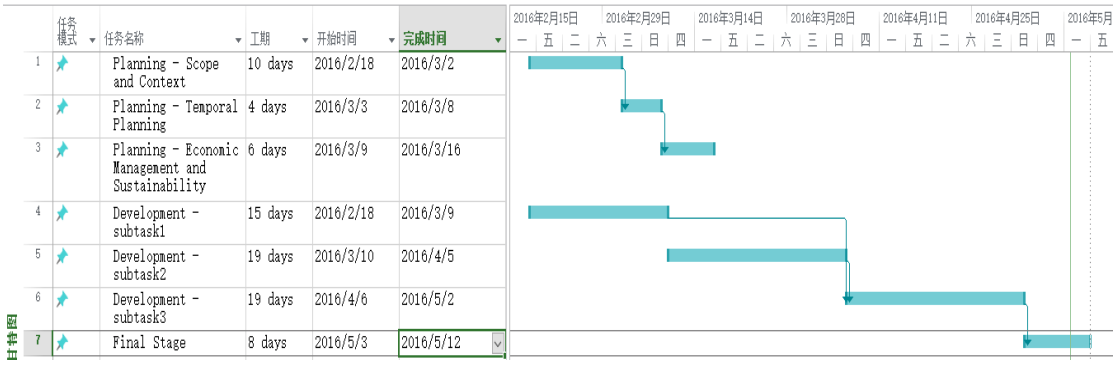


Figure 6 Gantt Chart of Final Timetable

2.3.2 Actual Expense

Since the whole project was extended, the cost on human resource increased. Meanwhile, the extension of the project also brings the increase on amortizations of software resource and hardware resource. The actual expense on human resource, hardware resource and software resource will be explained below.

Resource	Price(Euro)	Useful Life	Amortization(Euro)
Linux Fedora	0.00	3 years	0.00
Microsoft Windows 10	99.95	3 years	11.11
PyCharm 5 Community Edition	0.00	3 years	0.00
XShell 5 (Free for school/home use)	0.00	3 years	0.00
XFtp 5 (Free for school/home use)	0.00	3 years	0.00
Microsoft Office 2013	179.90	3 years	19.99
Adobe Reader	0.00	3 years	0.00
Total	279.85		31.10

Table 7 Actual Cost on Software Resource

Resource	Price(Euro)	Useful Life	Amortization(Euro)
Linux Server(with GPU)	20181.00	8 years	840.88
Lenovo Thinkpad E440	1225.00	5 years	81.67
Total	3729.00		922.55

Table 8 Actual Expense on Hardware Resource

The increase on the expense of human resource is mainly about the Software programmer and Software Tester, while the workload of program manager does not increase. The details about the expense on human resource is shown in Table 9.

Resource	Working Time(hours)	Price (euros/hour)	Cost(euros)
Program Manager	80	50.00	4000.00
Software Programmer	360	40.00	14400.00
Software Tester	160	35.00	5600.00
Total			24000.00

Table 9 Actual Expense on Human Resource

Combing the actual expense on hardware resource, software resource and human resource, the actual cost of this project is shown in Table 10.

Resource	Expense
Hardware Resource	922.55
Software Resource	31.10
Human Resource	24000.00
Total	24953.65

Table 10 Actual Expense of the whole project

2.4 Reasons of derivations

Usually, the derivations of the project mainly comes from two reasons, which were unexpected at the beginning of the project.

2.4.1 Training Time

The first reason is, the training of neural network took much more time than I expected. This system is a computing-intensive one and the training of neural network is slower than I expected. Before this project, I had no experience about the training of deep neural network. I used to build some simple neural network whose training only takes hours. However, for deep neural network, the training procedure will take days even weeks. Meanwhile, after the some iterations of training, sometimes I found the parameters that control the training is inappropriate, which means I need

to adjust the parameters and train it again, which took more time than I expected.

2.4.2 Code implementation

The other reason is about the coding implementation. The building of one of the subsystem, neural image caption generator is more difficult than I expected. At the beginning, I tried to build that subsystem based on an open-source software which is published in a top conference paper. However, after I tried about a month, I still didn't succeed. The code of that open source software is incomplete, which made it almost impossible for others to replicate or build their own software based on that. So, I have to turn to other solutions. This trying took too more time than I expected. Although I expected that the code implementation would be difficult, the extent of difficulty is still out of my expectation.

2.5 Sustainability analysis

In this part, I will analyze the sustainability of this project from different point of view: economy, society and environment.

2.5.1 Economic Sustainability

All the cost of this project, including hardware cost, software cost, and human resource cost as well as budget control method have been detailed described in the previous sections.

Due to the extension of the whole project, the cost on every resources increased while the main rise comes from the expense on human resource. Since I tried my best to use open-source software or free software as much as possible and share the Linux server with other teams, I have cut my expense on software resource and hardware resource as much as I can. However, due to many unexpected reasons, the cost of the whole project is beyond the budget.

The cost of this project may not be competitive due to the human resource that is put into it. Since the lack of experience in this area (machine translation), programmer may take longer time compared with those who are skilled in this area. However, programmer is trained well in this project. He now has a deeper understanding about Deep Learning and machine translation, which is very helpful for his future and the development of technology. So, if I take a look at the big picture, the cost is still competitive. Also, with trained programmer who is good at this area, I can replicate this project with lower cost on human resource and less time and even achieve better performance, which enhances its economic sustainability.

The whole project is set in the framework of WMT16[3]. As shown in previous sections, human resources are distributed to different tasks in accordance with their importance and difficulty. For example, much more time was devoted to integration of different results coming from two baseline systems.

2.5.2 Social Sustainability

With the globalization of economy, machine translation is more needed to aid the communication between different countries and cultures. The development of machine translation technology will definitely reduce the obstacles in the intercultural communication and accelerate the development of economy. My project, multimodal machine translation system, which is more about experiment, may not be employed in real life directly, however, building of this kind of system will help developers understand the shortage of current machine translation technology. This research project will accelerate the improvement of machine translation.

Undoubtedly, machine translation will take over some jobs of human translators. However, at least now, human translators cannot be replaced completely. There is still big room for the improvement on accuracy and fluency of machine translation system. This improvement is based on the bilingual corpus with good quality, which can only be provided by human translators.

2.5.3 Environmental Sustainability

All the resources needed have been listed in the *2.2 Resources Used* section.

The laptop and most software are used in all stages of this project. The server and some software (like Xftp, Xshell, Pycharm) are used at Development stage and final stage. This project does not generate toxic substance. However, due to the fact that this is a computing intensive program, this whole project may consume a lot of electricity. The power of laptop is about 100W while the server's power is about 1000W. The total consumption of electricity is about 470 kWh, meaning 298.92kg carbon dioxide. In order to save the electricity, I am going to cooperate with other students who need server, too. We are going to share the same server given the performance of that server is still satisfying. This method can save energy and money, and protect environment as well.

Almost all of our resources can be reused in other computer science classes or daily life. For example, Microsoft Office is a set of very useful office automation tools. Pycharm can be used by students who want to study python. Server and laptop can be used in almost any other application after this project.

[3] http://www.statmt.org/wmt16/multimodal-task.html

Furthermore, like I said in the previous part *2.5.2 Social Sustainability*, this research project is part of the advanced research about machine translation, which will facilitate the flow of all kinds of resource among different countries so as to lower the cost of manufacturing environmentally and economically. In this big vision, this project is also environmental sustainable.

2.5.4 Sustainability Assessment

Here is the sustainability matrix assessed according to the sustainability Report.

Sustainable?	**Economic**	**Social**	**Environmental**
Planning	Economic Viability [0:10]	Improved Quality of Life [0:10]	Resource Analysis [0:10]
Assessment	8	10	9
Outcomes	Final Cost Versus Forecast[-10:10]	Impact on Social Environment [-10:10]	Resource Consumption [-10:10]
Assessment	-5	10	0
Risks	Adapting to changes of Scenery[-20:0]	Social damage [-20:0]	Environment damage [-20:0]
Assessment	-2	0	0
Total	20		

Table 11 Sustainability Assessment

3 Basic Theories

In this part, related theories about this program will be introduced. Neural networks and algorithms described here are the basic components of this project.

3.1 From Perceptron to Neural Network

Perceptron is the most basic neural network proposed by Frank Rosenblatt in 1957 (Rosenblatt 1957). Actually, perceptron can be seen as a binary classifier. The architecture of a basic perceptron is shown in Figure 7.

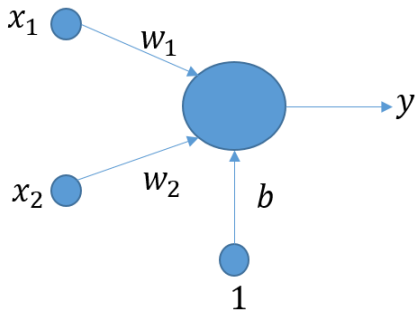


Figure 7 perceptron

In fact, perceptron built the model (3)

$$y = f(x) = act(W^T x + b) \quad (3)$$

where act is activating function, W is weight vector and b is bias. Activating function is usually $tanh$, $sign$ or $sigmoid$.

In order to get an expected classifier, an ideal weight vector W and corresponding bias b is needed. Given the training data, weight vector and bias will be adjusted so as to get the minimum error.

For linearly separable data, perceptron performs well, however, it cannot divide linearly inseparable data. In order to solve this problem, several perceptrons are put together so as to form a Multi-Layer Perceptron (MLP), which is also the most basic neural network.

Multi-layer perceptron can be divided into three parts, input layer, hidden layer(s) and output layer, as shown in Figure 8. Input layer is the units which accept the input directly without more computation. Output layer is the units that generate the final output. Hidden layers the units that are between input layer and output layer. There can be more than 1 hidden layer in neural network.

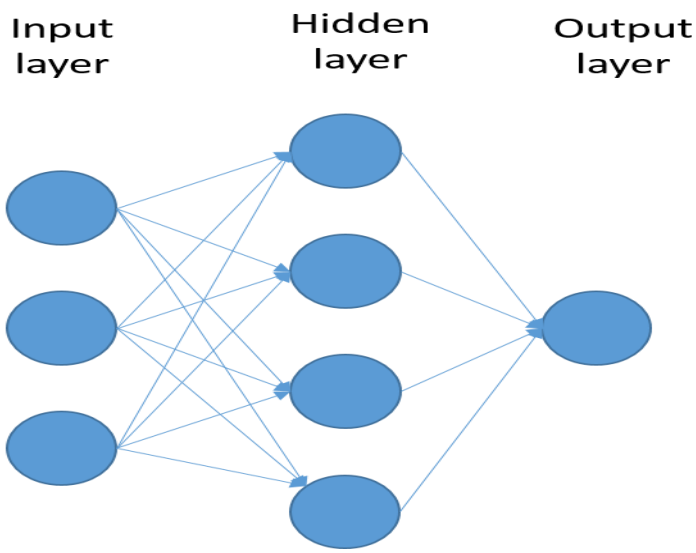


Figure 8 Multi-Layer Perceptron

The multi-layer perceptron is not perfect. The main problem in MLP comes from its connections. It is a fully connected neural network, which brings too many parameters, limiting the depth of the network and the number of units on every layer. For example, if the input is an image with the size of 256x256, and 256 units are on the hidden layer. Then there will be $256^3 = 16{,}777{,}216$ weight parameters at least. This problem limits the development and application of neural network.

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is one kind of feedforward neural network. The invention of Convolutional Neural Network is inspired by the visual mechanism of human's brain. Convolutional Neural Networks are widely used in image recognitions, video analysis and natural language processing (LeCun and Bengio 1995).

Three key concepts are the key to the success of Convolutional Neural Network, which overcomes the disadvantage of Multi-Layer Perceptron.

Local Receptive View: Similarly to human eyes, which usually focus on a local part of the image while looking around, in the Convolutional Neural Network, units in hidden layers are only connected to some of units in the previous layer, unlike multi-layer perceptron where units in hidden layers are fully connected to the previous layer. Local Receptive Field lowers the number of weight parameters that need to be trained.

Weight Sharing: In the human nerve system, neurons in the same tissue usually function in the same way with similar architecture. Similarly, in the convolutional neural network, units in the same layer share weight parameters, which decrease the number of parameters waiting to be trained greatly and improve the efficiency.

Pooling: When a person close his or her eyes, it is impossible to recall all the details of the things he or she just saw, which, however, does not hinder his or her memory about those things. Things have been "compressed" in our memory and the loss of details does not affect the function of our brains. That is how pooling came. After convolution calculation, features are pooled so as to decrease the size of feature map.

In this part, a typical Convolutional Neural Network (Figure 9), will be used to illustrate the theory of Convolutional Neural Network.

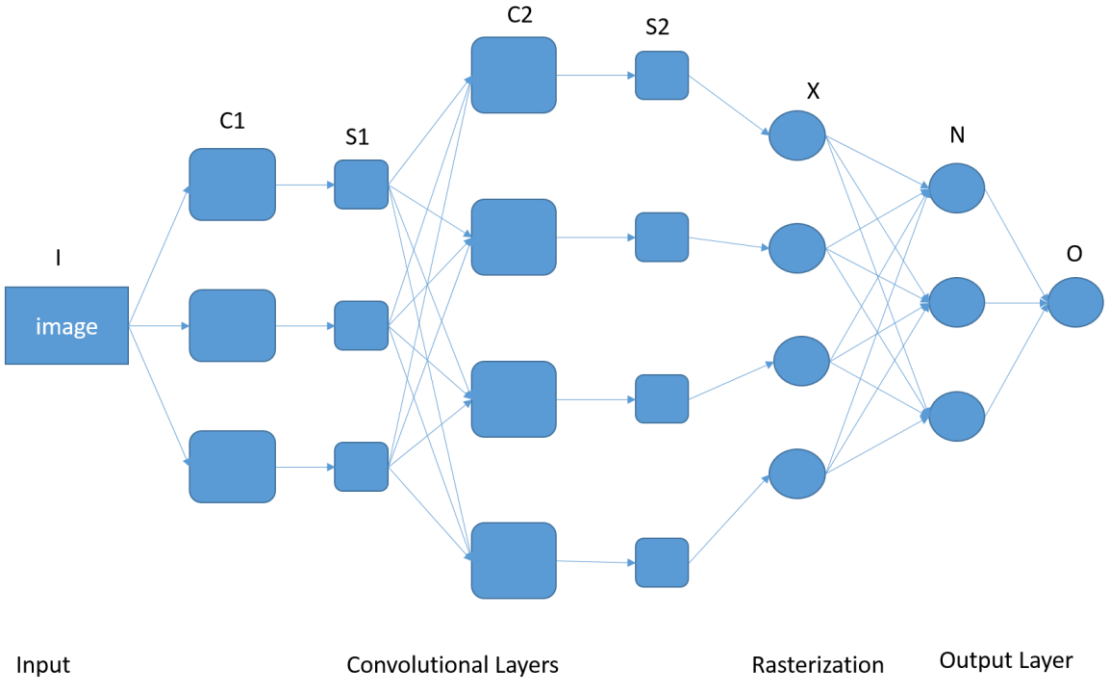


Figure 9 Convolutional Neural Network Structure

As shown in Figure 9, in most cases, Convolutional Neural Network can be divided into four parts:

Input Image (I): this image can be a grayscale image, can also be a RGB image. Also, if the input image is an RBG image, then there will be three input images.

Convolutional Layers: multiple convolution layers (C) and down-sampling layers (S). Convolution layers will calculate the convolution of the weight parameters and the output from the previous layer while the output of convolution layers will be down-sampled (pooling) by down-sampling layers. The output in this part is called feature maps.

The convolution layers work according to (4)

$$f(x) = act(\sum_{i,j}^{n} w_{(n-i)(n-j)} x_{ij} + b) \quad (4)$$

For example, supposing the output from the previous layer is a 5x5 image (or feature map), and the kernel size of convolution layer is 3X3. The output feature map is also in the size of 3x3. In the first step, the kernel will convolute with the most upper left 3x3 submatrix of input matrix to get the value of the first unit in the output feature map. In the second step, the kernel will convolute with another submatrix (overlapped with the first one), shown in Figure 10.

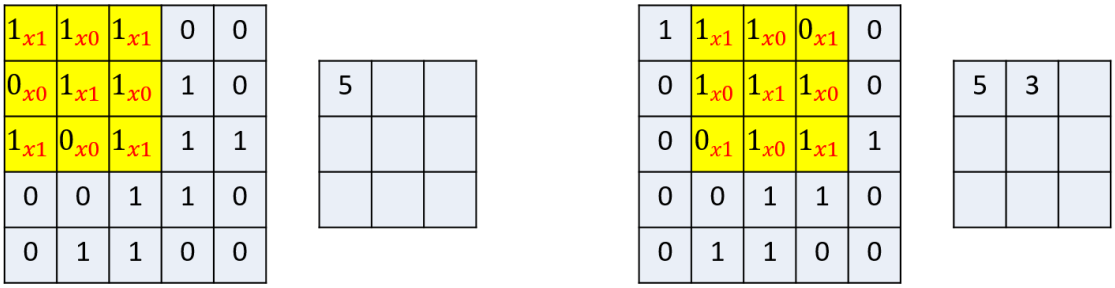


Figure 10 Process of Convolution Layers

The down-sampling layers compress the feature maps by pooling, which improve the neural network by decreasing the number of features as well as avoiding overfitting. The size of pooling is usually 2x2, common pooling methods include max pooling, choosing the maximum in the pool as the output, and mean pooling (calculating the mean value in the pool as output). The pool in every step does not overlap with each other, as shown in Figure 11.

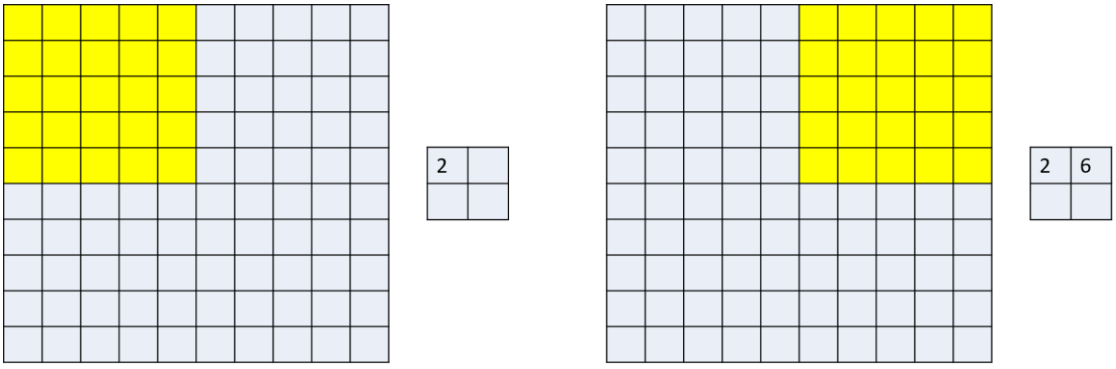


Figure 11 Process of down sampling layers

Rasterization (X): feature maps are actually matrices, the rasterization layer will unfold the matrices and convert them into vectors.

Output Layer (N and O): traditional neural network. This part will give the final output of convolutional neural network.

Training algorithm of Convolutional neural network works in the same way as Back-Propagation algorithm in traditional neural network. The error is traced back to every weight parameter. Weight parameters are updated in order to decrease the cost of the whole neural network.

The definition of the cost function on the whole convolutional neural network is shown in (5):

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 \quad (5)$$

Where $h_{W,b}(x)$ is the output of convolutional neural network give the input x, while y is the supervised signal of convolutional neural network. Since $J(W,b)$ is a non-convex function, gradient decreasing method may converge to a local optimum, but in practice, gradient decreasing methods usually generate satisfying output. It is emphasized that all the parameters should be

initialized randomly instead of being initialized to zero. It will make the contributions to the error completely equal by initialing all parameters to zero, which will make it impossible to converge.

In every iteration of gradient decreasing methods, W and b will be updated according to the formulas (6) and (7)

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{6}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{7}$$

α is learning rate.

The Back-Propagation algorithm is described below.

I. Forward computing. Calculate the output L_2, L_3 … until the output layer L_n;

II. For every unit in output layer, residuals are calculated according to (8):

$$\delta_i^{(n)} = -\left(y_i - a_i^{(n)}\right) * g'(z_i^{(n)}) \tag{8}$$

For $\mathrm{l} = \mathrm{n}-1, \mathrm{n}-2, \mathrm{n}-3 \ldots 2$:

$$\delta_i^{(l)} = \left((W^{(l)})^T \delta_i^{(l+1)}\right) * g'(z_i^{(l)}) \tag{9}$$

III. The final partial derivatives:

$$\nabla_{W^{(t)}} J(W, b) = \delta_i^{(l+1)} (a^l)^T \tag{10}$$

$$\nabla_{b^{(t)}} J(W, b) = \delta_i^{(l+1)} \tag{11}$$

With the methods described above, convolutional neural network can be trained and approaches the optimum in the way of back-propagation. In practice, a training dataset can be divided into several batches and the neural network will be trained over and over again so as to get the best performance.

3.3 Recurrent Neural Network

It's impossible for a human to think without prior knowledge. Our current understanding of this world is based our previous experience and knowledge. Information from the past can help us comprehend and make decisions now. Furthermore, our current experience will stay in our mind and aid us in the future. The holding of information from the past is one of the most important keys of our evolution.

However, it seems like a major weakness for a traditional neural network that it does not hold previous information. For example, if you want to analysis the emotion of every sentence in a passage, it is nearly impossible for a traditional neural network to complete it, since it cannot use information from the previous sentence, which is vital to this task.

Recurrent Neural Network solved this problem. It has been widely applied in Natural Language

Processing (NLP) and performs very well on multiple tasks (Mikolov, Karafiát et al. 2010, Sutskever, Vinyals et al. 2014).

Recurrent Neural Networks are used to handle sequence data. In a traditional neural network, units from different layers (input layer, hidden layers, and output layer) are fully connected while the units in the same layer are not connected. For traditional neural network, output is only relevant to current input. However, in Recurrent Neural Network, output are not only related to the current input, but also previous input, that is why it is called 'recurrent'. Recurrent Neural Network will store the information from previous input and use it to calculate the current output, which means the connections among units in the same hidden layer exist and the input of hidden layer includes the output of the input layer at this moment and the output of hidden layer at last moment. Theoretically, Recurrent Neural Network can handle sequence data in any length. But in practice, it is usually assumed that the output is only relevant to a few previous inputs.

Figure 12 is a typical Recurrent Neural Network.

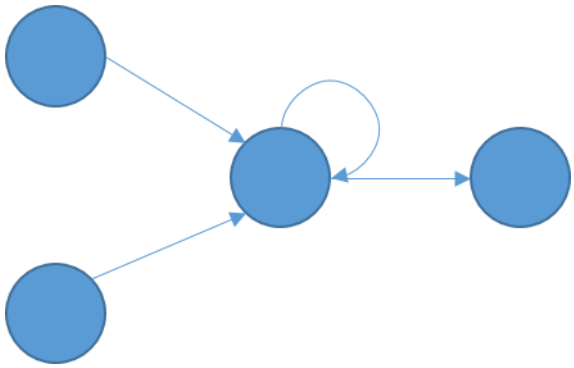


Figure 12 Typical Recurrent Neural Network

The connection between the units in the same hidden layer made Recurrent Neural Network magical. However, Recurrent Neural Network can also be viewed from a different angle. It can be seen as a very deep neural network if you unfold the iterations as shown in Figure 13.

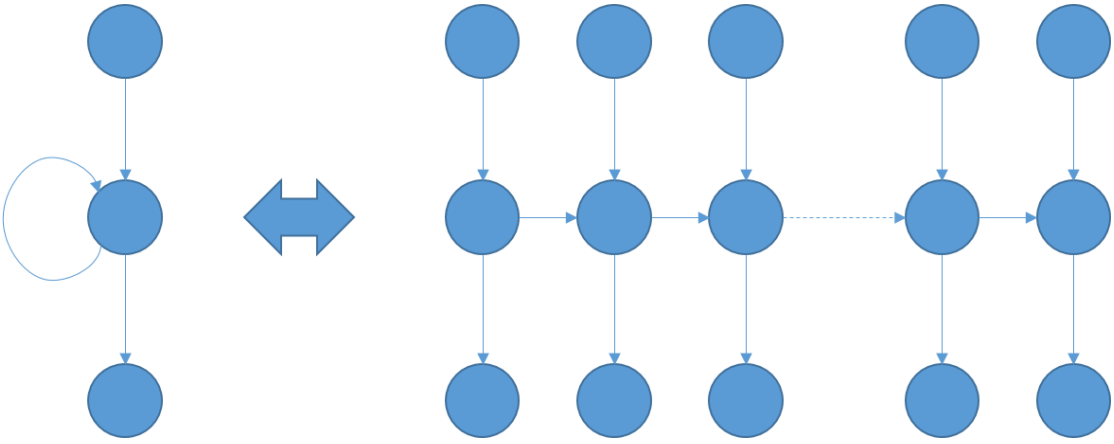


Figure 13 Unfolding Recurrent Neural Network

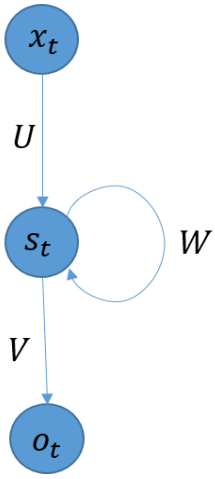


Figure 14 Workflow of RNN

Recurrent Neural Network contains input units, output units and hidden units. As shown in Figure 14, input of Recurrent Neural Network is $\{x_0, x_1, \dots, x_t, x_{t+1}, \dots\}$, output of Recurrent Neural Network is $\{o_0, o_1, \dots, o_t, o_{t+1}, \dots\}$ while the state of hidden unit is $\{s_0, s_1, s_2, \dots, s_t, s_{t+1}, \dots\}$. In the workflow shown in Figure 14, a flow of information originating from the input unit go into the hidden unit while another flow of information originating from the hidden unit go into the hidden unit itself. In some cases, Recurrent Neural Network will break the limit by leading the information from output unit into the hidden unit, which is called back projections. Hidden unit also takes the state from the previous hidden unit as input, meaning hidden units can be interconnected or self-connected.

x_t is the input at time $t, t = 1,2,3$ …, s_t is the state of hidden unit at time t. s_t is based on the input at the current time and the hidden state of the previous time, which is $s_t = f(Ux_t + Ws_{t-1})$ while f is usually tanh. o_t is the output at time t.

s_t can be regarded as memory unit which contains all the hidden state in previous time. o_t is only related to current s_t.

In a traditional neural network, parameters in the same layer are not shared. However, similar to that of Convolutional Neural Network, units of Recurrent Neural Network in the same layer also shares parameters (U, V, W).

Recurrent Neural Network has been proved to be successful in various tasks of Natural Language Processing, such as Part-of-Speech tagging and machine translation. The most successful Recurrent Neural Network model is Long Short-Term Memory, which will be explained in the 3.4 Long Short Term Memory.

3.4 Long Short Term Memory

One of the biggest advantages of Recurrent Neural Network is that it can use previous information for the current task, for example, in Machine Translation, a Recurrent Neural Network can use the information from the words that have already been translated for the translation of current word. Recurrent Neural Network can be very useful if it can achieve that all the time. But in some situation, it cannot use previous information, especially distant previous information, effectively.

In Recurrent Neural Network, hidden unit is used to hold the information in the past. In practice, the hidden state is usually represented as a length-fixed vector, which means, no matter how much information it has gained in the past, all of that has to be compressed into that length-fixed vector. As the accumulation of information, loss of information seems to be unavoidable. Information gained long time ago may be abandoned inadvertently, which brings the problem of long term dependencies. Although in theory, a Recurrent Neural Network can handle this long term dependencies problem. But it is too hard to learn the needed parameters. Bengio et al dug in depth about this problem (Bengio, Simard et al. 1994).

The space that will be used to contain the previous information is not unlimited, meaning we cannot hold all the information without screening, so we need a mechanism that can help us decide what to remember, what to forget and what to output. And that is how Long Short-Term Memory came into being.

Long Short-Term Memory, usually referred as LSTM, a special Recurrent Neural Network, can learn the long term dependencies. Long Short Term Memory is brought by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber 1997). Long Short Term Memory performs extraordinarily well on numerous tasks and is being widely used.

The main difference between the traditional Recurrent Neural Network and Long Short-Term Memory is the most basic unit in the neural network. In traditional Recurrent Neural Network, the most basic unit is perceptron, but the basic unit in Long Short-Term Memory is a memory cell with three gates, input gate, forget gate and output gate, as shown in Figure 15.

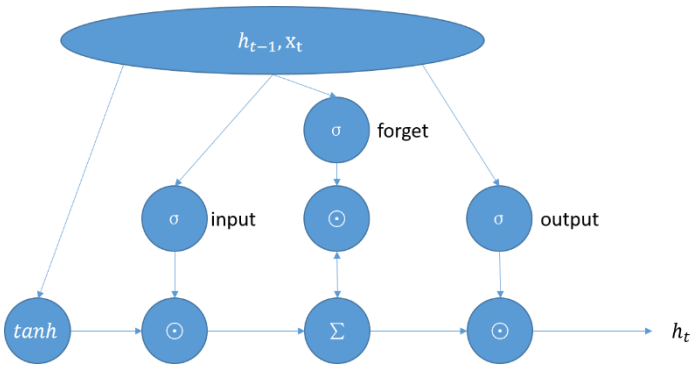


Figure 15 Memory Cell in LSTM

Memory cell enables Long Short –Term Memory also take the previous state of the memory cell into consideration, apart from the output from previous memory cell and current input, which gives it the advantage when predicting the long term dependencies. Forget gate, input gate and output gate work together to decide the final output.

The output h_t will be decided by (12) (13)

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o) \quad (12)$$

$$h_t = o_t * tanh(C_t) \quad (13)$$

where o_t is the output of output gate, which will decide if the state in memory cell will be output.

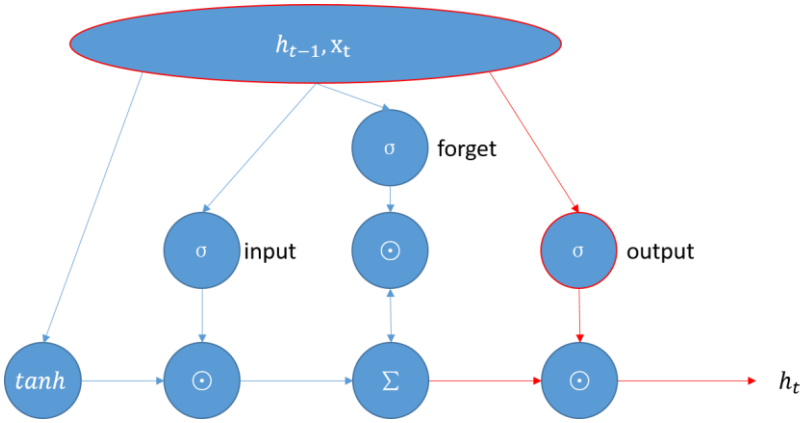


Figure 16 output gate of LSTM

The inner state of the memory cell C_t will be decided by (14)

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \quad (14)$$

where f_t is the output of forget gate, i_t is the output of input gate, C_{t-1} is the output from previous memory cell and $\widetilde{C}_t$ is the intended new value of memory cell.

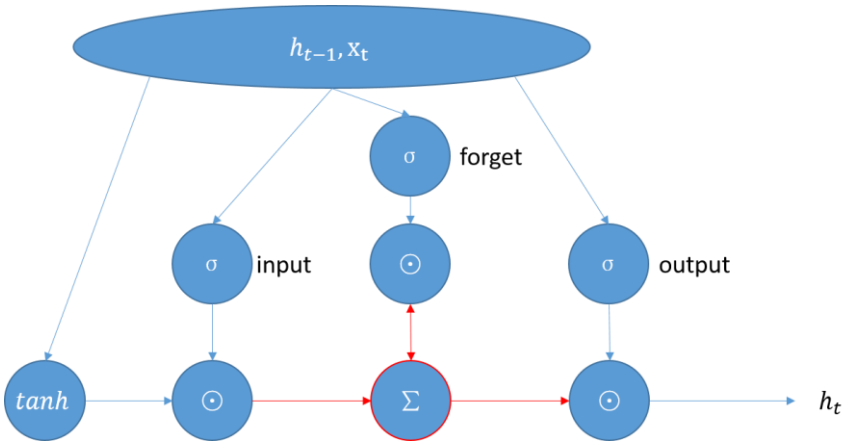


Figure 17 inner state of LSTM

The output of input gate and intended new value of Long Short-Term Memory will be decided by (15) (16)

$$i_t = \sigma(W_i\,[h_{t-1}, x_t] + b_i) \quad (15)$$

$$\widetilde{C}_t = tanh(W_C\,[h_{t-1}, x_t] + b_C) \quad (16)$$

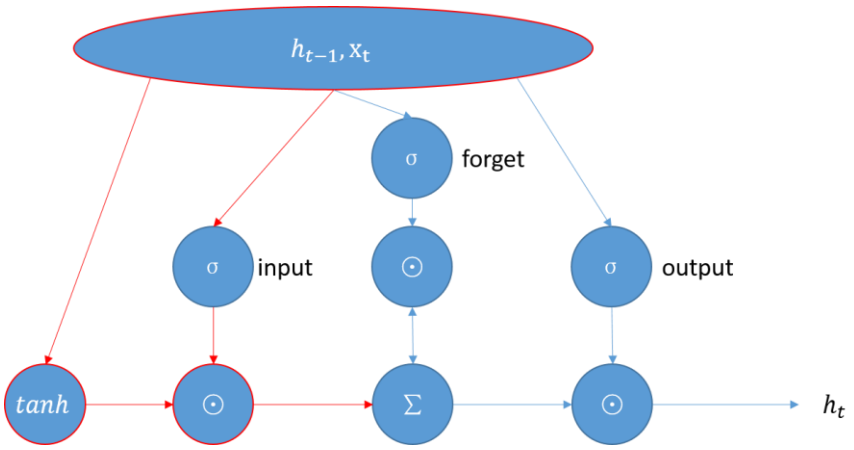


Figure 18 input gate of LSTM

Forget gate in memory cell will be decided by (17)

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (17)$$

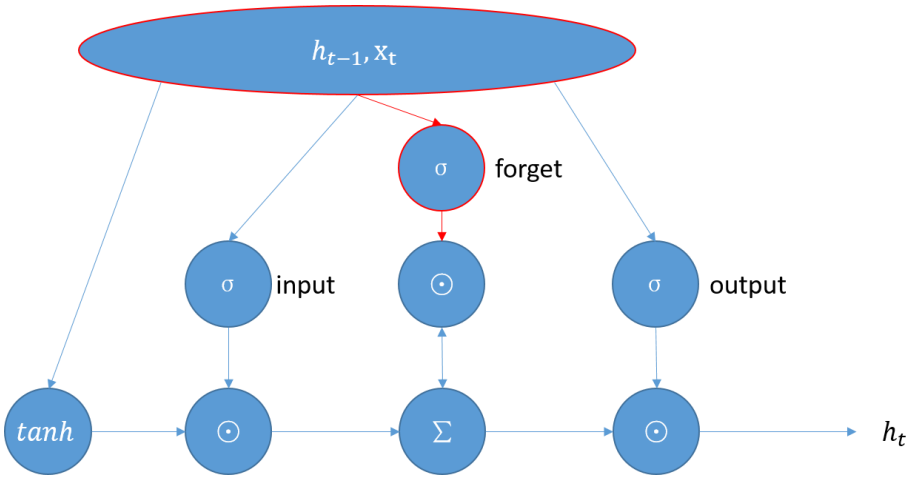


Figure 19 forget gate of LSTM

3.5 N-gram Language Model

Language Model builds the model on an infinite set of string in natural language composed by a finite number of words. The challenge of modeling natural language comes from the infiniteness of sentences and variable length of string. N-gram language model is proposed by IBM in 1992 (Brown, Desouza et al. 1992). Taking advantage of Markov process, n-gram language model assumes that the appearance of the current word is only related to the previous $n-1$ words. The probability of the whole sentence is the product of the probabilities of the appearance of each

word, which can be represented as (2).

Commonly used n-gram language model is trigram language model ($n = 3$), meaning the appearance of the next word is only conditioned on the previous two words.

A language model can be used for scoring sentence, meaning assessing the probability of the appearance of the sentence according to the training corpus. Language model will first be trained on the corpus so as to get the statistic information about n-grams.

The probability of each n-gram can be got by simply count the frequency.

$$P(w_i|w_{i-n+1}w_{i-n+2}\dots w_{i-1}) = \frac{count(w_{i-n+1}w_{i-n+2}\dots w_{i-1}w_i)}{count(w_{i-n+1}w_{i-n+2}\dots w_{i-1})} \quad (18)$$

However, due to the infiniteness of sentence, this naive method may encounter the problem of sparse data, as well as making it hard to deal with n-gram that does not appear in the training corpus. So, additional methods, like linear interpolation and discounting method will also be used to keep the n-gram model running.

3.6 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method that can be used for regression and classification. Support Vector Machine is known for its ability to perform classification and regression tasks on linearly inseparable data with kernel tricks, which will map the data that is linearly inseparable in low-dimension space to high-dimension space where it can be separated by hyperplane.

As shown in Figure 20, in a 2D space, circles and crosses represent data in two different classes. If it is needed to classify the data, the best solution is to find a line that can divide the whole 2D space into two parts, each of which contains only one class of data.

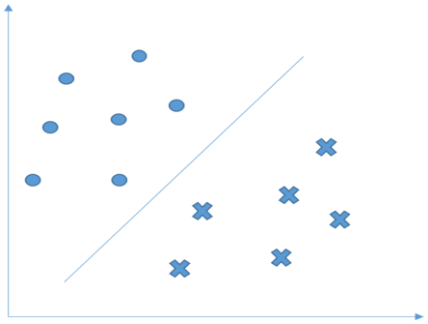


Figure 20 hyperplane in 2D space

Furthermore, the longer the distance between the closest cross (or circle) and the plane, the better it performs. In mathematics, the line that divide the space into two parts can be represented as (19)

$$w^T x + b = 0 \tag{19}$$

Now, it is assumed that for the closest circle (or cross), it is on the line $\mathrm{w}^{\mathrm{T}} x + b = 1$ or $\mathrm{w}^{\mathrm{T}} x + b = -1$, and the job is to find the w that maximizes the distance between the data and line, as shown in Figure 21, which is why this algorithm is called Support Vector Machine, the closest circle or the cross is so called “support vector”.

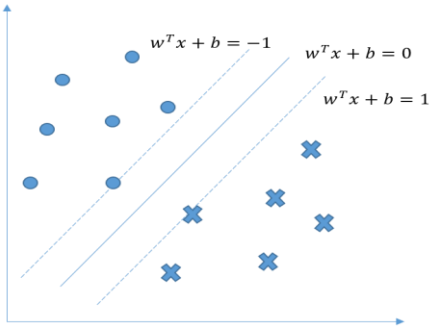


Figure 21 "Support Vector" of SVM

The goal can be represented as (20)

$$\max_{\mathrm{w,b}} \frac{|w^T x+b|}{||w||} = \max_{w} \frac{1}{||w||} = \min_{w} \frac{1}{2}||w||^2 \tag{20}$$

Lagrange multiplier method can be used to get the solution of w.

$$\mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{21}$$

where $\alpha_i$is Lagrange multiplier.

So the model of support vector machine can be represented as (22)

$$\mathrm{f(x)} = (\sum_{i=1}^{n} \alpha_i y_i x_i)^{\mathrm{T}} x + b = \sum_{i=1}^{n} \alpha_i y_i < x_i, x > +b \tag{22}$$

For linearly inseparable data, Support Vector Machine can use kernel functions to map the data from low-dimension to high-dimension space first, then a hyper space in high dimensional space will be found to separate the data, as shown in Figure 22， which means the model of Support Vector Machine is represented as (23)

$$\mathrm{f(x)} = \sum_{i=1}^{n} \alpha_i y_i < x_i, x > +b = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i, x) + b \tag{23}$$

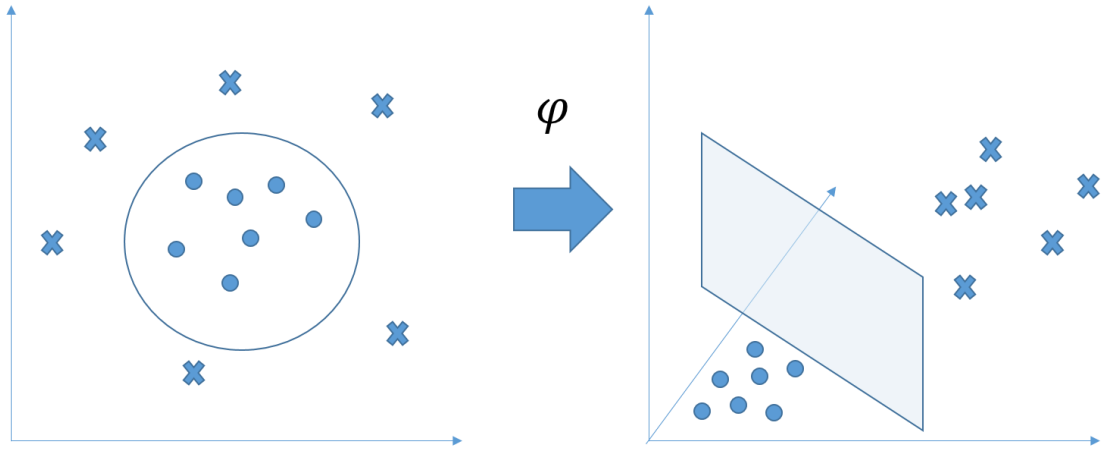


Figure 22 Kernel Function of Support Vector Machine

3.7 BLEU

Bilingual Evaluation Understudy (BLEU) is proposed by IBM in 2002. BLEU is an algorithm used for assessing the quality of the text translated by a machine. BLEU completed the assessment by comparing the candidate text generated by the machine and reference text written by human. It is natural that the more similar text translated by computer is to a reference text provided by human, the better its quality is (Papineni, Roukos et al. 2002). BLEU is an automatic evaluating method with very low cost, which is why it is being widely used today.

Usually, BLEU evaluation is usually conducted at sentence level. But if it is needed to assess the quality of a document or a corpus, BLEU will first be calculated for every sentence and their average will be the final BLEU for that sentence or corpus.

Borrowing the idea from n-gram language model, BLEU evaluate the quality of sentence by comparing the n-gram in the translated sentence and reference sentences. First, BLEU will calculate the precision of n-gram:

$$\mathrm{p_n} = \frac{\sum_{C\epsilon candidates}\sum_{n-gram\epsilon C} count_{clip}(n-gram)}{\sum_{C\epsilon references}\sum_{n-gram\epsilon C} count(n-gram)} \tag{24}$$

$$\mathrm{count_{clip}}(n-gram) = \min\{count(n-gram), \max\_ref\_count(n-gram)\} \tag{25}$$

$\mathrm{count}(n-gram)$ is the number of n-gram appears in the sentence while $\mathrm{count_{clip}}(n-gram)$ is the revised number of n-gram appears in the sentence, $\max\_\mathrm{ref\_count}(\mathrm{n-gram})$ is the maximum of n-gram appears in the reference sentence.

After getting the value of precision of every n-gram, BLEU can be computed as (26) :

$$\mathrm{BLEU} = \mathrm{BP} * \exp(\sum_{n=1}^{4} w_n \log p_n) \tag{26}$$

w_n is the weight of each n-gram. BP is brevity penalty:

$$BP = \begin{cases} 1, & if\ c > r \\ e^{1-\frac{r}{c}}, & if\ c \leq r \end{cases} \tag{27}$$

c is the length of candidate translation and r is the length of reference sentence.

4 Program Implementation

In this part, I will first describe the architecture of the whole program and then describe the implementation details of three modules: neural machine translator, neural image caption generator and evaluation module.

4.1 Program Architecture

The aim of this project is to build a multimodal neural machine translation system, which takes an image and more captions in the source language (English), to generate a caption in the target language (German). This project can be viewed from two different angles.

First, it can be seen as to generate captions in the target language (German) from image aided by the features from captions in the source language (English); it can also be regarded as a mission to translate the caption from the source language (English) to target language (German) supported by the features from raw images.

From this point of view, the whole program is designed like this, captions will be generated from two different directions, the first caption in the target language will be generated directly from the image, which will be completed by neural image caption generator, and the second caption in the target language will be generated based on the captions in the source language, which will be finished by neural machine translator.

After that, an evaluation module will be used to assess the quality of these two different translations, both fluency and correctness will be taken into consideration. This evaluation module will rescore these two sentences and decide which one is the best translation that will be the final output.

The architecture of the whole system is shown in Figure 23.

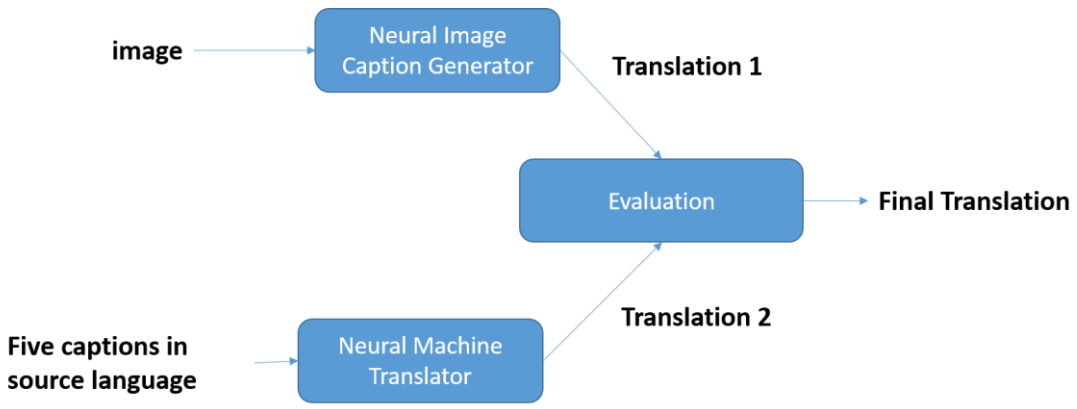


Figure 23 System Architecture

In Neural Image Caption Generator, a Convolutional Neural Network will be used to extract features from the images and features extracted will be used by a Long Short Term Memory sentence generator to generate captions.

In Neural Machine Translator, an attention-based neural machine translator will be used to translate the captions in the source language to target language.

In the evaluation module, an n-gram language module, being trained on reference corpus, will be used to assess the quality of two translated sentences. After that, a Support Vector Machine based Re-Scorer will be used to give the final score for each translated sentence, scores coming from neural machine translator and neural image caption generator, as well as scores from language module will all be taken into consideration.

4.2 Neural Machine Translator

Neural Machine translator used here is an encoder-decoder model with attention-based mechanism. For the encoder part, a bidirectional Recurrent Neural Network is used to get the representation of the source sentence, after that, an attention-based decoder is used to generate the sentence in the target language. In this part, I followed the method proposed by Bahdanau and Cho (Bahdanau, Cho et al. 2014).

4.2.1 Bidirectional Recurrent Neural Network Encoder

As I stated in 1.4.1 Neural Machine Translation, the major problem of Recurrent Neural Network in Encoder-Decoder architecture used in machine translation is that the Recurrent Neural Network will encode the whole sentence into one length-fixed vector, which brings the loss of information especially when it tries to deal with long sentences. Bidirectional Recurrent Neural Network addressed this problem by encoding the source sentence into a group of vectors, whose number is same as that of words in the source sentence.

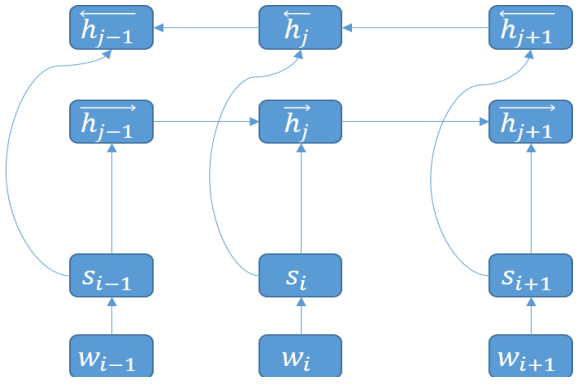


Figure 24 Bidirectional RNN Encoder

A Bidirectional Recurrent Neural Network consists of a forward Recurrent Neural Network and a backward Recurrent Neural Network. The forward Recurrent Neural Network will read the sentence in the source language forward, namely from beginning to end, while the backward Recurrent Neural Network will read the sentence backward, namely from end to beginning. The concatenation of the output from the forward Recurrent Neural Network and the backward Recurrent Neural Network is the representation of the whole sentence.

While encoding the source sentence, every word in the source sentence will be first represented as a one-hot vector (also known as 1-of-K representation), which contains no information about any kind relationship among words. After that, word embedding techniques $\mathrm{s_i} = Ew_i$ will convert the one-hot vector to a continuous vector that is usually in a lower dimensional space and contains more information about the relationship among words.

The bidirectional Recurrent Neural Network works in the way shown in (28)(29)(30)

$$\overrightarrow{h_j} = \varphi(\overrightarrow{h_{j-1}}, s_j) \tag{28}$$

$$\overleftarrow{h_j} = \varphi(\overleftarrow{h_{j+1}}, s_j) \tag{29}$$

$$\mathrm{h_j} = [\overrightarrow{h_j}, \overleftarrow{h_j}] \tag{30}$$

$\overrightarrow{h_j}$ is the output of the forward Recurrent Neural Network which has read the first j words in the source sentence. $\overleftarrow{h_j}$ is the output of the backward Recurrent Neural Network which has read all the words after j-th position. $\mathrm{h_j}$, also referred to as annotation vector, is the concatenation of $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$.

It can be seen that $\overrightarrow{h_j}$ represents the information before the j-th word while $\overleftarrow{h_j}$ represents information after j-th word. So the concatenation of $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$, namely $\mathrm{h_j}$, represents the whole sentence. It will be easier for a Recurrent Neural Network to recall recent information. So, by computing the annotation vector that represents the whole sentence for the position of each word, I avoid the loss of information about context. That is also why the output of a bidirectional Recurrent Neural Network is a better representation of the original sentence.

So, in the methods described above, a sentence in the source language has been encoded into a group of vectors, which save the context information and whose number is the same as that of words in the source sentence. This mechanism avoids the loss of information that traditional Recurrent Neural Network encountered when it compresses the source sentence into a length-fixed vector.

4.2.2 Attention-based Decoder

Unlike traditional encode-decoder neural machine translator, as shown in Figure 25, an attention-

based decoder is used in decoding. Annotation vectors got in the last part will be the input of this decoder and a small neural network, attention mechanism, will decide the weight that will be put into each annotation vector.

The core of this decoder is still a Recurrent Neural Network, where the internal state of which is decided by the annotation vectors, the previous word and its previous internal state, as shown in (31)

$$z_i = \varphi'(c_i, z_{i-1}, u_{i-1}) \tag{31}$$

c_i is the expectation of annotation vectors at position i.

The expectation of the annotation vectors at position i is a weighted average the annotation vectors. The weight of each annotation vector is decided by a small neural network, which is called attention mechanism. This small neural network has a single hidden layer and a scalar output. It takes the previous hidden state of recurrent neural network decoder, as well as one of the annotation vector as input and will be repeated for every annotation vector. This process can be described as (32) (33) (34).

$$e'_{i,t} = f(z_{i-1}, h_t) \tag{32}$$
$$a_{i,t} = softmax(e'_{i,t}) \tag{33}$$
$$c_i = \sum_t a_{i,t} * h_t \tag{34}$$

With internal state z_i got from the recurrent neural network, the probability distribution of the next word can be computed in the following way.

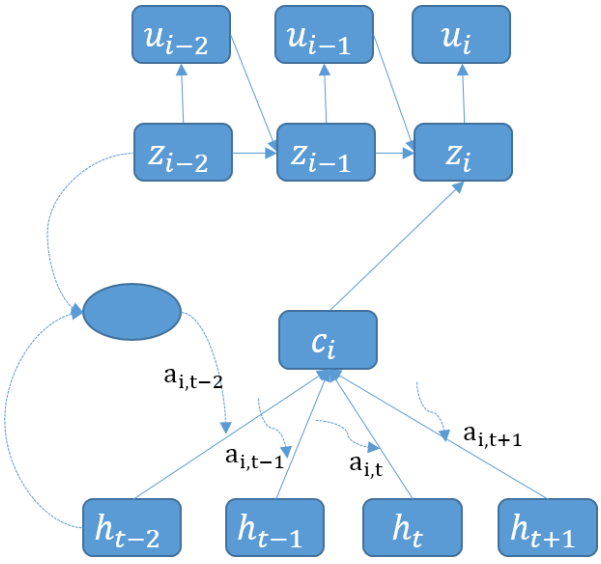


Figure 25 attention-based decoder

Given the internal hidden state z_i, each candidate word can be scored by (35)

$$e(k) = {w_k}^T z_i + b_k \quad (35)$$

w_k is the word vector of the word k.

The reason I choose dot product is that, according to its definition, the closer two vectors, the bigger their dot product. So, the word that is the closest to the internal hidden state of Recurrent Neural Network will get the highest score in this process.

With the score that I got for each word, I can compute the probability of each word using softmax (Briddle 1990) as shown in (36).

$$\mathrm{p}(\mathrm{w_i} = \mathrm{k}|\mathrm{w_1}, \mathrm{w_2}, \ldots, \mathrm{w_{i-1}}) = \frac{\exp(\mathrm{e(k)})}{\sum_j \exp(e(j))} \quad (36)$$

In this way, I can get the word for the i–th position by sampling the distribution. After getting the word for the i-th position, I can repeat this process until the generation of the word <eos> (end-of-sentence).

Following the procedures described above, I can get the translation of the image description from its corresponding captions in the source language. The best translation of this subsystem will be used as the input of the evaluation module.

4.3 Neural Image Caption Generator

In the neural image caption generator, every image is first sent into a Convolutional Neural Network to extract the features from raw images. After that, features extracted will be used to generate the sentence. Features extracted by a Convolutional Neural Network can be seen as another representation of the raw image, which makes it easier to express and compute. In this part, I followed the method proposed by Vinyals et al (Vinyals, Toshev et al. 2015).

4.3.1 Convolutional Neural Network Image Feature Extractor

Convolutional Neural Network is widely used to extract features from the image. In this part, in order to maximize the performance of the system, I used a Convolutional Neural Network proposed by Simonyan & Zisserman (Simonyan and Zisserman 2014), which yields the best performance on ILSVRC 2014 classification competition. Furthermore, this Convolutional Neural Network has also shown great capability of scene classification by transfer learning (Donahue, Jia et al. 2013).

The Convolutional Neural Network used here has 16 weight layers. The final softmax layer was removed so that a 4096 dimensional vector can be used to represent the raw image. Every image will first be resized to 224x224 so as to fit the Convolutional Neural Network. The input of the feature extractor is 224x224 RGB images and the output is a 4096 dimensional vectors which can

be regarded as a digitized representation of the raw image itself. The 4096 dimensional vector got in this part is the input of the Long Short – Term Memory based sentence generator, which will be described in 4.3.2 LSTM-based Sentence Generator.

4.3.2 LSTM-based Sentence Generator

The requirement of this subsystem is to generate the sentence that describes image in the target language based on the features extracted from the image. The input of this subsystem is 4096 dimensional vectors got from the convolutional neural network image feature extractor, the output is a sentence with variable length. So, it is natural to generate words one by one and the generation of the next word is based on the previous ones that have been generated and the features extracted from the image. So, this task can be represented as (37).

$$\arg\max_{\theta} \log \mathrm{p}(S|I,\theta) = \arg\max_{\theta} \sum_{t=0}^{N} \log \mathrm{p}(S_t|I,\theta,S_0,S_1,\dots,S_{t-1}) \tag{37}$$

I is the feature from the image, θ is the parameter, N is the length of the sentence, S is sentence generated and S_i is the i-th word in the sentence.

A Long Short – Term Memory used to model the process, $\mathrm{p}(S_t|I,\theta,S_0,S_1,\dots,S_{t-1})$. Image feature is the initial input of this Long Short –Term Memory and words will be generated along the way. In order to keep the Long Short – Term Memory going, every word generated will be encoded in a vector with same dimensionality of the image feature, which is 4096. The process can be seen more clearly if I unfold the Long Short Term Memory in Figure 26.

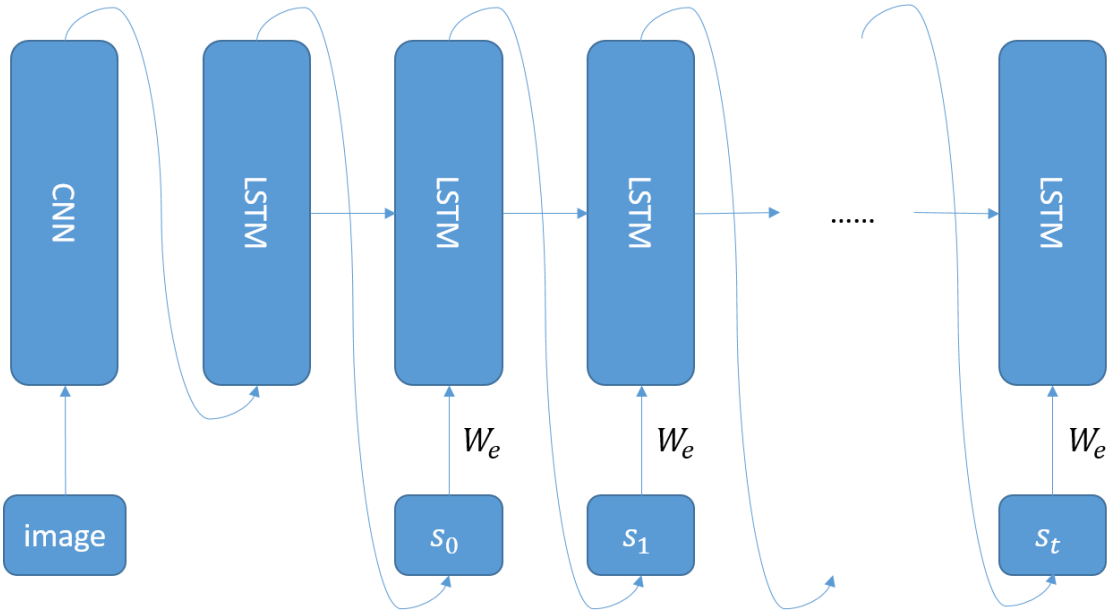


Figure 26 Workflow of LSTM based Sentence Generator

The whole procedure can also be represented as (38) (39) (40):

$$\mathrm{x}_{-1} = CNN(I) \tag{38}$$

$$\mathrm{x}_{\mathrm{t}} = W_e S_t \tag{39}$$

$$p_{t+1} = LSTM(x_t) \quad (40)$$

The image is used only once at the first time, W_e is the word embedding matrix and S_t is a one-hot vector. By sampling p_t, I can get the sentence that describes the content of every image.

The Long Short Term Memory used here is slightly different from the one explained in 3.4 Long Short Term Memory. The LSTM used here is computed by (41) (42) (43) (44) (45) (46)

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \quad (41)$$
$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \quad (42)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \quad (43)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \quad (44)$$
$$m_t = o_t \odot c_t \quad (45)$$
$$p_{t+1} = softmax(m_t) \quad (46)$$

$\odot$ represents the product with a gate value, σ is sigmoid function while h is tangent, multiple W matrices are trained parameters.

At the same time, multiple techniques are used to generate the sentence, such as beam search and sampling.

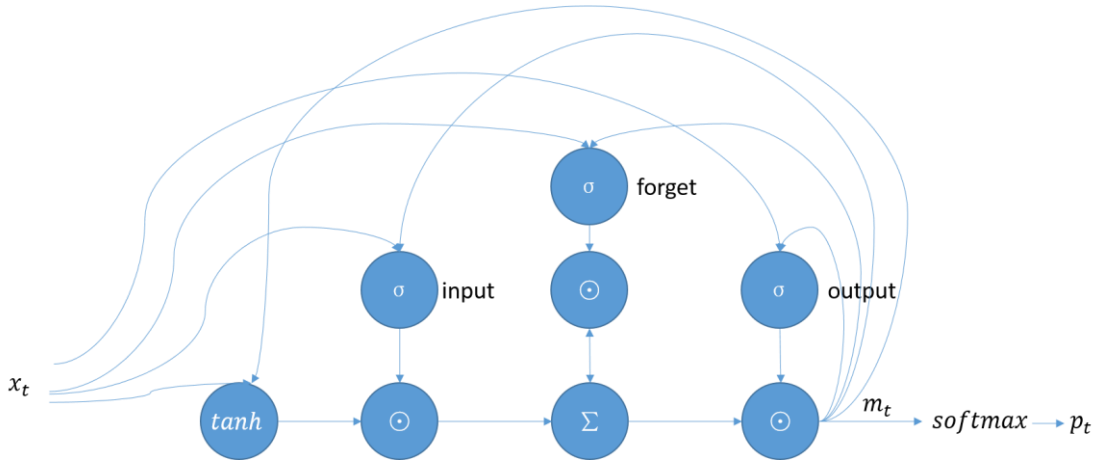


Figure 27 LSTM-based Sentence Generator

4.4 Evaluation Module

Currently, I already got at least two sentences from the previous subsystem, one from neural machine translator, and another one from neural image caption generator. So, the task of this module is to evaluate which sentence is the best translation or the best description of the image. So, this module can also be regarded as a rescoring process.

When I evaluate the quality of sentence generated, I need to consider the correctness and fluency at the same time. Some sentences generated may be correct enough, but sound not so 'human', which means it lacks fluency. While others sentences may be very fluent, but it does not describe the image, which means it lacks correctness.

The previous two subsystems, neural image translator and neural image caption generator, not only generate sentences, but also the scores of them. This score mainly focuses on the correctness. More importantly, the scores of sentences generated by different system are not in the same coordinate system. For example, even if two sentences generated by neural image translator and neural image caption generator respectively have the same score from their original systems, this does not mean that they are in the same quality.

So, in order to evaluate the quality of the sentences generated for the same image better, an n-gram language model was led in, which will evaluate the fluency of the sentences. The n-gram model will also give a score to the sentences generated, which mainly focus on the fluency.

Currently, every sentence has two scores, one from the model that generates it and another from language model. In order to standardize, every sentence is given another zero-value score. If the sentence is generated by neural machine translator, then its score from neural image caption generator is zero. Similarly, if a sentence is generated by neural image caption generator, then its score from neural machine translator is zero. So, every sentence has three scores now. Since the final result will be evaluated with BLEU, I will try to choose the sentence with the highest BLEU based on the three scores of each sentence.

The relationship among the three scores and the final BLEU is unclear. It is possible that they are not linearly correlated. So, a Support Vector Machine, which is known for modeling nonlinear relationship, will be used to find the best translation, the sentence with highest BLEU value.

In the training stage, the three score of every sentence, along with the BLEU value of the sentence will be used to train the Support Vector Machine. The Support Vector Machine will try to find the relationship among the three scores and the final BLEU. The BLEU cannot be gotten without the reference. So in the test stage, when the reference is unknown, the Support Vector Machine will predict the BLEU based on the three scores. Sentence with the highest BLEU prediction will be chosen as the best description sentence for that image.

5 Experiments

5.1 Data preparation

Currently, extensive research about Image Description has been conducted on English-language dataset. However, in this project, Image Description is not limited to English. In order to train and test my multimodal neural machine translation system, Multi30k dataset (Elliott, Frank et al. 2016) is used in the training and experiment.

Multi30k extends Flickr30k (Young, Lai et al. 2014) in two different ways. In the first way, a subset of English description is created by selecting one English description for every image, and German translation for that subset is provided by professional translators without showing the raw images. In the second way, five German image description are provided through crowdsourcing according to their English versions and independently from each other.

WMT16 provided two tasks of multimodal machine translation shared task. For the first task, the organizers provided Multi30k dataset that extends Flickr30k in the first way, namely one caption in the source language (English) and one caption in the target language (German) for every image. For the second task, the organizers provided Multi30k that extends Flickr30k in the second way, namely five captions in the source language (English) and five captions in the target language (German) for every image.

The difference of the dataset for two tasks is not only the number of captions provided, but also the similarity between the German sentence and English sentence in the same pair. For the first task, any English – German pair is a parallel pair, which means the dataset can be regarded as a parallel corpus. For the second task, each German – English pair is a comparable pair.

The aim of both tasks is to generate just one caption for the image in the target language (German). I mainly focus on the second task but I still run parts of my system on the first task so as to compare the performance.

For both tasks, the organizers provided 29000 tuples for training, 1014 tuples for validation, and 1000 tuples for test. The splits of dataset in two tasks are the same.

	Training	**Validation**	**Test**
size	29000	1014	1000

Table 12 Size of dataset

5.2 Results and Analysis

The experiment is conducted in three steps. In the first step, Neural Machine Translator will be trained on the datasets provided by two tasks respectively and I will compare their performance. In the second step, Neural Image Caption Generator will also be trained on two different corpora and I will compare the performance as well. In the last step, I will choose the combination of two subsystems that are trained for the same task and perform better to merge their results through a Support Vector Machine based Evaluation module so as to improve the final results.

5.2.1 Neural Machine Translator

In the first step, two Neural Machine Translators were trained for two tasks separately. For the first task, every tuple contains one English-German pair while for the second task every tuple contains five English-German pairs. With the same number of tuples used for training, neural machine translator used for the second task got larger training corpus, which consists of 145000 English-German pairs, while the neural machine translator used for the first task 29000 English-German pairs.

System	Number of English-German pairs for training	BLEU
Neural Machine Translator for task 1	29000	25.50
Neural Machine Translator for task 2	145000	9.39

Table 13 Neural Machine Translator Performance

Usually, with the increased size of the corpus used for training, performance of the system should be better, more or less. However, experiment shows that neural machine translator didn't perform better as the size of the training corpus went up, even more, the performance became worse.

As stated before, the difference between the corpora in two tasks is not only size, but also the alignment between sentences in the same pair. For the first task, organizers provided a parallel corpus, while for the second task, organizers provided a comparable corpus.

The alignment of a parallel corpus is usually at sentence level. A parallel corpus can be used to explore how express the same meaning in different languages. For any sentence pair in the parallel corpus, sentences in that pair contain almost the same information without increase or loss of details. Commonly used parallel corpus including EUROPARL (Koehn 2005), PKU Chinese-English Parallel Corpus (Chang 2005).

But the alignment of a comparable corpus is usually at topic level, which means, although being in the same domain, sentences in the same pair may vary in length and details. For example, news in the same topic published by different agencies in different languages can be considered a comparable corpus. Commonly used comparable corpus includes The International Corpus of

English (ICE) [4] and The Lancaster-Oslo/Bergen Corpus (LOB Corpus)[5].

For example, here is a sample from a parallel corpus provided in the first task:
English: Two young, White males are outside near many bushes.
German: Zwei junge weiße Männer sind im Freien in der Nähe vieler Büsche.

And here is a sample from the comparable corpus provided in the second task.
English: Two friends enjoy time spent together.
German: Leute vor einer Gartentür im sommelichen Garten, der noch sommerlicher wirkt, weil es ganz unabhängig von dem Bild Winter ist bei uns.

The dissimilarity between sentences in the same pair caused the ineffectiveness of Neural Machine Translator. The Encoder-Decoder Architecture supported by attention mechanism used here performs very well on a parallel corpus, but is not robust enough to learn from a comparable corpus which contains much more noise.

5.2.2 Neural Image Caption Generator

Two Neural Image Caption Generators are also trained on two different datasets provided by the organizers. Since organizers provided five captions in the target language (German) for each picture in the second task, the dataset for second task is as big as five times of the dataset for the first task. The training dataset for the first task contains 29000 image-caption pairs while there are 145000 image-caption pairs for the training of task 2.

System	Number of image-caption pairs for training	BLEU
Neural Image Caption Generator for task 1	29000	2.41
Neural Image Caption Generator for task 2	145000	8.19

Table 14 Neural Image Caption Generator Performance

Experiment shows that the performance of Neural Image Caption Generator is better with the increased size of training dataset. Neural Image Caption Generator does not take the captions in the source language (English) as input, so the difference of information contained by sentences in the same pair does not affect the performance of Neural Image Caption Generator.

Compared with the results of systems that uses same technology but generates English captions, this system does not perform very well even when it is trained on the second task dataset, which may be caused by complex German morphology that results into larger vocabulary.

[4] http://ice-corpora.net/ice/
[5] http://www.helsinki.fi/varieng/CoRD/corpora/LOB/

5.2.3 Whole System

Although the result of Neural Machine Translator on the first task is really good, due to the bad performance of Neural Image Caption Generator on the first task, the incorporation of the systems trained for the first task will make little or no improvement. So, I decided to merge the output of two systems that trained for the second task, whose performance is close to each other and it is very possible that the incorporation of these two systems will make a real difference.

In order to validate the effectiveness of my Support Vector Machine based Evaluation Module, I also rescored all the sentences with n-gram language model only.

System	BLEU
Neural Machine Translator	9.39
Neural Image Caption Generator	8.19
Merged by Trigram Language Model only	9.65
Merged by 5-gram Language Model only	9.73
Merged by SVM with Trigram Language Model	11.52
Merged by SVM with 5-gram Language Model	11.52

Table 15 System Performance on task 2

As Table 15 shows, merging the result from Neural Machine Translator and Neural Image Caption Generator will improve the performance of the whole system with all the methods that have been tried. If I only use the score from n-gram language model for choosing the best sentence, I can get improvement but it is very limited compared with those incorporated by Support Vector Machine based Evaluation model. Support Vector Machine improved the system better since it considered correctness and fluency at the same time while a language model only considered fluency but neglected the correctness of the sentence based on the picture.

It is also found that by increasing n in the n-gram language model, the improvement is not so obvious especially when a Support Vector Machine is introduced. The emphasis on the fluency cannot improve the system greatly.

By integrating the Neural Machine Translator, neural image caption generator, language model and Support Vector Machine based Evaluation Module, I completed a system that can complete the task of multimodal machine translation and the introduction of the Support Vector Machine based Evaluation system improved the performance of the whole system.

5.3 Results Show

In this part, I will show some result of the test set.

Ein Mann mit Sonnenbrille

Ein Hund läuft über eine Wiese

Ein kleiner Junge spielt mit
einem kleinen Jungen.

Ein Mann in einer leuchtend gelben
Weste regelt einen Essenswagen .

Eine Mutter freut sich auf einem
Laufband im Freien mit ihrer
Angelrute .

Figure 28 Results show - 1

Menschen auf einer Straße

Zwei Männer arbeiten auf einer Baustelle

Ein Baseballspieler wirft einen Ball.

Menschen auf einem verschneiten Berg

Ein Mann sitzt auf einem Felsen

Figure 29 Results show - 2

6 Conclusion

6.1 Contribution

With the development of society and progress of technology, it is not a dream for human any more to translate text from one language to another. Traditionally, researchers usually focus on pure text machine translation. With the recent great progress of deep learning techniques that have been widely used in computer vision and speech recognition, more and more scientists began to do research in multimodal translation, whose input or output is multimodal text.

The unique architecture of Recurrent Neural Network made it successful in many fields, such as image captioning, language modeling and machine translation. Two baseline system of this project, is based on Recurrent Neural Network. The main contribution of this project are to integrate current system so as to complete the task of multimodal machine translation and improve the performance of the whole system.

More specifically, this project proposed a set of methods that are capable of completing the multimodal machine translation task, which is to generate a caption in the target language (German), given the image itself and corresponding one or more image captions in the source language (English) . By utilizing and improving the current techniques in image captioning and machine translation, two captions will be generated from different directions, one from image directly by a Convolutional Neural Network image feature extractor followed by a LSTM-based sentence generator, another one from the captions in the source language by an attention-based Encoder-Decoder Recurrent Neural Network. Sentence with the best quality will be the final output.

This report also proposed a method of re-scoring by using a support vector machine, scores from the n-gram language model and two baseline system, neural machine translator and neural image caption generator, will be used to predict the quality of the sentence. The rescoring process will consider the correctness and fluency at the same time. Experiments also show that the Support Vector Machine – based sentence rescorer did improve the performance of the whole system.

6.2 Future Work

However, the theory and methodology proposed in this project is far from perfect and still need to be improved. In the future, the improvement of the whole system can be considered in the following aspects:

The analysis the current image captioning and machine translation technologies: more analysis should be done about the current image captioning and machine translation systems so as to find

the weakness and advantages in them. This insight will bring inspirations to improve them, and the performance of the whole system will be improved along the way.

The combination of image captioning and machine translation: currently, image and captions in the source language are put into different neural networks, and the output with the best quality will be the final output. However, the ideal solution should be to put the image (or image features) and captions in the source language into the same neural network, whose output is the caption needed in the target language. The deep combination of machine translation and image captioning technology is still waiting for the future study.

Acknowledgements

At the end of this report, I would like to appreciate all the professors and students who helped me in the past four years!

This report is completed with the supervision from Prof. Marta Ruiz Costa-jussà and Prof. Lluís Padró Cirera. Their rigorous attitude towards education and research, animate thinking and rich research experience gave me a very deep impression and influenced me a lot. I would like to offer my most sincere gratitude to them.

I would also like to thank all the professors who taught me in the past four years. Thank you very much for sharing knowledge with me, which gives me solid basis and illuminates my road ahead.

I also want to thank my family who supports me and takes care of me as always. It is impossible for me to make any accomplishment without their help.

At last, I would like to thank all the professors in committee for reading and evaluating my project report.

Reference

Antony, P. (2013). "Machine translation approaches and survey for indian languages." Computational Linguistics and Chinese Language Processing **18**(1): 47-78.

Auli, M., et al. (2013). Joint Language and Translation Modeling with Recurrent Neural Networks. EMNLP.

Bahdanau, D., et al. (2014). "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473.

Bengio, Y., et al. (2003). "A neural probabilistic language model." J. Mach. Learn. Res. **3**: 1137-1155.

Bengio, Y., et al. (1994). "Learning long-term dependencies with gradient descent is difficult." Neural Networks, IEEE Transactions on **5**(2): 157-166.

Briddle, J. S. (1990). "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters."

Brown, P. F., et al. (1992). "Class-based n-gram models of natural language." Computational linguistics **18**(4): 467-479.

Chang, B. (2005). "Chinese-English parallel corpus construction and its application."

Cho, K., et al. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259.

Cho, K., et al. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078.

Cho, S. J. K., et al. (2015). "On Using Very Large Target Vocabulary for Neural Machine Translation."

Donahue, J., et al. (2013). "Decaf: A deep convolutional activation feature for generic visual recognition." arXiv preprint arXiv:1310.1531.

Elliott, D., et al. (2016). "Multi30K: Multilingual English-German Image Descriptions." arXiv preprint arXiv:1605.00459.

Farhadi, A., et al. (2010). Every picture tells a story: Generating sentences from images. Computer Vision–ECCV 2010, Springer**:** 15-29.

González, L. P. (2014). "Multimodality in Translation and Interpreting Studies: Theoretical and Methodological Perspectives." A Companion to Translation Studies: 119-131.

Hochreiter, S. and J. Schmidhuber (1997). "Long short-term memory." Neural computation **9**(8): 1735-1780.

Kalchbrenner, N. and P. Blunsom (2013). Recurrent Continuous Translation Models. EMNLP.

Kiros, R., et al. (2014). Multimodal neural language models. Proceedings of the 31st International Conference on Machine Learning (ICML-14).

Kiros, R., et al. (2014). "Unifying visual-semantic embeddings with multimodal neural language models." arXiv preprint arXiv:1411.2539.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. MT summit.

Kulkarni, G., et al. (2013). "Babytalk: Understanding and generating simple image descriptions." Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(12): 2891-2903.

Kuznetsova, P., et al. (2012). Collective generation of natural image descriptions. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Association for Computational Linguistics.

Kuznetsova, P., et al. (2014). "TREETALK: Composition and Compression of Trees for Image Descriptions." TACL **2**(10): 351-362.

LeCun, Y. and Y. Bengio (1995). "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks **3361**(10): 1995.

Li, S., et al. (2011). Composing simple image descriptions using web-scale n-grams. Proceedings of the Fifteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics.

Lin, T.-Y., et al. (2014). Microsoft coco: Common objects in context. Computer Vision–ECCV 2014, Springer**:** 740-755.

Luong, M.-T., et al. (2015). "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025.

Mao, J., et al. (2014). "Explain images with multimodal recurrent neural networks." arXiv preprint arXiv:1410.1090.

Mikolov, T. (2012). "Statistical language models based on neural networks." Presentation at Google, Mountain View, 2nd April.

Mikolov, T., et al. (2010). Recurrent neural network based language model. INTERSPEECH.

Mitchell, M., et al. (2012). Midge: Generating image descriptions from computer vision detections. Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics.

Papineni, K., et al. (2002). BLEU: a method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para, Cornell Aeronautical Laboratory.

Simonyan, K. and A. Zisserman (2014). "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556.

Sutskever, I., et al. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems.

Vinyals, O., et al. (2015). Show and tell: A neural image caption generator. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Xu, K., et al. (2015). "Show, attend and tell: Neural image caption generation with visual attention." arXiv preprint arXiv:1502.03044.

Yang, Y., et al. (2011). Corpus-guided sentence generation of natural images. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Young, P., et al. (2014). "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions." Transactions of the Association for Computational Linguistics **2**: 67-78.